AMENDMENTS TO THE CLAIMS

The following listing of the claim(s) will replace all prior versions and listings of claim(s) in the application.

Claims 1-5 (canceled)

6. (Currently Amended.) A computer implemented method of detecting vulnerabilities in a pre-existing source code listing, said source code listing having a listed sequence of expressions, each expression including a set of operands and operators to transform values of the operands, said listed sequence of expressions having an inherent control flow indicative of the run-time execution of the expressions and an inherent data flow indicative of the run-time transformations of operand values, said source code listing further having routine calls, said routine calls including arguments with which to invoke a routine, said arguments including expression-references and operand-references to computer files, said source code listing being stored in a computer-readable medium, said computer implemented method comprising the acts of:

executing computer instructions to analyze the source code listing to create computer models of said control flow to indicate the run-time sequence in which routine calls will be invoked and to create computer models of said arguments for the routine calls using a flow insensitive analysis, wherein said control flow models include a control flow graph, and wherein each of said models of arguments is stored in computer memory and specifies pre-determined characteristics about and a range of possible values for the corresponding argument as a result of said source code expressions;

executing computer instructions to use said computer models of said control flow in order to determine a run-time sequence of execution of a pair of routine calls by traversing the control flow graph backwards, said pair of routine calls having a first routine call and second routine call in which execution of the first routine call precedes execution of said second routine call;

executing computer instructions to determine ~~if~~ whether a second routine to be executed has ~~an~~ second argument with a corresponding modeled range of possible of values that includes a reference ~~referring~~ to a file that is also within a corresponding modeled range of possible values for ~~referred to by~~ ~~an~~ first argument of the first routine to be executed, so that a possibility of the first and second arguments referring to the same file is determined even when said expression-references and operand-references to computer files for said first and said second arguments are lexically dissimilar;

executing computer instructions ~~and if so~~ to identify said sequence as a race condition vulnerability; and

generating a report that is viewable by a user and that identifies the race condition vulnerabilities, so the user may modify the source code listing to address the vulnerability if desired.

7. (Previously Presented.) The method of claim 6 further including the act of executing computer instructions to analyze the source code listing to create computer models of said data flow to indicate the run-time transformations of operand values and including the act of using data flow models to resolve the expression-references and operand-references to computer files in the first and second routine calls to detect whether both routines refer to the same computer file.

8. (Cancelled.)

9. (Currently Amended.) A system for detecting vulnerabilities in a pre-existing source code listing, said source code listing having a listed sequence of expressions, each expression including a set of operands and operators to transform values of the operands, said listed sequence of expressions having an inherent control flow indicative of the run-time execution of the expressions and an inherent data flow indicative of the run-time transformations of operand values, said source code listing further having routine calls, said routine calls including arguments with which to invoke

a routine, said arguments including expression-references and operand-references to computer files, said source code listing being stored in a computer-readable medium, said system comprising:

computer-executable instructions on a computer-readable medium to analyze the source code listing to create computer models of said control flow to indicate the run-time sequence in which routine calls will be invoked and to create computer models of said arguments for the routine calls using a flow insensitive analysis, wherein said control flow models include a control flow graph, and wherein each of said models of arguments is stored in computer memory and specifies pre-determined characteristics about and a range of possible values for the corresponding argument as a result of said source code expressions;

computer-executable instructions on a computer-readable medium to use said computer models of said control flow to determine a run-time sequence of execution of a pair of routine calls by traversing the control flow graph backwards, said pair of routine calls having a first routine call and second routine call in which execution of the first routine call precedes execution of said second routine call;

computer-executable instructions on a computer-readable medium to determine ~~if~~ whether a second routine to be executed has ~~an~~ second argument with a corresponding modeled range of possible of values that includes a reference ~~referring~~ to a file that is also within a corresponding modeled range of possible values for ~~referred to by~~ ~~an~~ first argument of the first routine to be executed, so that a possibility of the first and second arguments referring to the same file is determined even when said expression-references and operand-references to computer files for said first and said second arguments are lexically dissimilar;

computer-executable instructions on a computer-readable medium ~~and if so~~ to identify said sequence as a race condition vulnerability; and

computer-executable instructions on a computer-readable medium to generate a report that is viewable by a user and that identifies the race condition vulnerabilities, so the user may modify the source code listing to address the vulnerability if desired.

10. (Previously Presented.) The system of claim 9 further including computer-executable instructions on a computer-readable medium to analyze the source code listing to create computer models of said data flow to indicate the run-time transformations of operand values and to use the data flow models to resolve the expression-references and operand-references to computer files in the first and second routine calls to detect whether both routines refer to the same computer file.

11. (Cancelled.)